UDC 004.056

# SECURE MULTI-PARTY COMPUTATION METHODS FOR CONFIDENTIAL BIG DATA ANALYTICS

**Bargsyan A.A.**
*master's degree, State engineering university of Armenia (Yerevan, Armenia)*

# МЕТОДЫ ЗАЩИЩЁННЫХ МНОГОПОЛЬЗОВАТЕЛЬСКИХ ВЫЧИСЛЕНИЙ ДЛЯ КОНФИДЕНЦИАЛЬНОЙ АНАЛИТИКИ БОЛЬШИХ ДАННЫХ

**Баргсян А.А.**
*магистр, Государственный инженерный университет Армении (Ереван, Армения)*

**Abstract**

This article explores secure multi-party computation (SMPC) as a foundational cryptographic approach for performing collaborative analytics on sensitive big data without compromising privacy. The study analyzes the architectural components, protocol mechanisms, and practical considerations for integrating SMPC into large-scale analytical systems. Key focus areas include data representation, secure aggregation, performance optimization, and interoperability with machine learning workflows. Through illustrative examples and technical evaluation, the paper highlights current limitations and emerging solutions for scalable, privacy-preserving computation. The findings offer insights into designing secure analytics pipelines suitable for real-world deployment across regulated and distributed environments.

**Keywords:** secure multi-party computation, confidential data, privacy-preserving analytics, Big Data, distributed protocols, secure aggregation, machine learning, data protection.

**Аннотация**

В статье рассматриваются методы безопасных многопартийных вычислений (SMPC) как криптографическая основа для конфиденциальной обработки больших данных в условиях распределённой аналитики. Исследуются архитектурные принципы, механизмы протоколов и практические аспекты внедрения SMPC в масштабируемые аналитические платформы. Особое внимание уделяется представлению данных, безопасной агрегации, стратегиям оптимизации производительности и интеграции с рабочими процессами машинного обучения. Приведённые примеры и технический анализ демонстрируют существующие ограничения и потенциальные решения в области защищённой вычислительной аналитики. Представленные результаты способствуют формированию надёжной и масштабируемой среды для анализа чувствительных данных в различных отраслях.

**Ключевые слова:** безопасные многопартийные вычисления, конфиденциальные данные, приватная аналитика, большие данные, распределённые протоколы, защищённая агрегация, машинное обучение, защита данных.

## Introduction

The growing reliance on large-scale data-driven systems has intensified concerns regarding data confidentiality, particularly in contexts where multiple stakeholders must jointly analyze sensitive datasets. Traditional approaches to secure analytics often require data centralization, which introduces significant privacy risks and regulatory challenges. As industries increasingly adopt distributed data processing across organizational or jurisdictional boundaries, ensuring privacy without compromising analytical capabilities becomes a critical objective. This tension is especially evident in sectors such as healthcare, finance, and public governance, where regulatory frameworks prohibit raw data sharing while demanding collaborative insights.

Secure multi-party computation offers a cryptographic paradigm that enables joint computation over private inputs without revealing individual data to participating entities. By allowing mutually distrusting parties to collaborate on data processing while preserving input confidentiality, SMPC provides a foundation for privacy-preserving analytics across decentralized infrastructures. Recent advances in cryptographic protocols, including secret sharing, homomorphic encryption, and oblivious transfer, have significantly improved the efficiency and scalability of such systems, making them more viable for integration with Big Data technologies.

This paper aims to examine the methodological foundations and practical implementation aspects of SMPC in the context of confidential analytics on large-scale data. The study explores protocol designs, system architectures, and deployment considerations necessary for real-world applications. It also analyzes performance trade-offs, compatibility with existing data infrastructures, and potential for integration with machine learning pipelines. The research contributes to the development of secure computational environments where privacy, accuracy, and scalability can coexist without the need for centralized trust.

## Main part

### Core protocol structure of secure multi-party computation

Secure multi-party computation protocols are designed to allow multiple participants to jointly compute a function over their private inputs while ensuring that no party gains access to the inputs of others. At the heart of these protocols lies the definition of a shared computational goal expressed as a function, typically decomposed into basic arithmetic or logical operations. Each party encrypts or encodes its data in a way that permits manipulation without exposing the raw values, enabling cooperative execution of the overall computation [1]. Such protocols are underpinned by foundational cryptographic mechanisms such as additive secret sharing, where data is split into fragments distributed among participants.

A typical computation proceeds in synchronized rounds, with each round consisting of local computation, message exchange, and reconstruction. The communication topology and trust assumptions determine whether protocols follow a semi-honest or malicious threat model. In semi-honest scenarios, parties follow the protocol but may try to infer hidden information; in malicious models, active deviation is anticipated and must be mitigated with verifiable computation steps. Protocols must also be resilient to latency and data loss in distributed networks, which is particularly important in big data settings where computation spans heterogeneous and geographically dispersed systems [2].

The pseudocode below presents a simplified implementation of an additive secret sharing protocol used to compute the sum of inputs from multiple parties without revealing their individual values.

```
# Simplified additive secret sharing for sum computation

import random

def share_secret(secret, n):
    """Split secret into n shares."""
    shares = [random.randint(0, 1000) for _ in range(n - 1)]
    final_share = secret - sum(shares)
    shares.append(final_share)
    return shares

def reconstruct(shares_list):
    """Reconstruct original value from all shares."""
    return sum(shares_list)

# Example: three-party computation
secret_inputs = [30, 50, 40]  # confidential inputs from three parties
n = len(secret_inputs)

# Each party shares their input
all_shares = [share_secret(secret, n) for secret in secret_inputs]

# Each party sums received shares for local partial result
partial_sums = [sum(party_shares) for party_shares in zip(*all_shares)]

# Reconstruct final result
result = reconstruct(partial_sums)
print("Final computed sum without revealing inputs:", result)
```

This example demonstrates the essential mechanics of additive secret sharing, where private inputs are decomposed into distributed shares and securely aggregated without disclosure. The protocol is lightweight and suitable for summation tasks or input averaging in collaborative environments [3]. While simplistic, it forms the foundation for more complex secure computations involving matrix operations, machine learning inference, or statistical analysis in privacy-sensitive domains.

**Data representation and encoding techniques in SMPC workflows**

The effectiveness of SMPC protocols in large-scale analytical systems heavily depends on how data is represented and encoded prior to computation. Unlike conventional processing, where raw values are directly accessible, SMPC systems require that inputs be transformed into protected formats that preserve both structure and operational compatibility. Data encoding must support modular arithmetic and be resilient to truncation, overflow, and rounding errors, especially in floating-point domains. This becomes particularly relevant in privacy-preserving statistical computations, where accuracy must be retained across decentralized operations [4].

Integer-based encoding schemes, such as fixed-point representation, are widely adopted due to their compatibility with arithmetic sharing mechanisms. These formats enable efficient addition and multiplication over finite fields or rings, allowing protocols to operate on encrypted or secret-shared data without the need for costly cryptographic conversions. Moreover, batch encoding strategies have emerged to improve throughput in high-dimensional datasets, enabling parallel computation over matrix-shaped data. Careful selection of modulus size and base precision is crucial to maintain both correctness and performance.

In addition to numeric representations, categorical and structured data pose specific challenges [5]. Common preprocessing steps-such as one-hot encoding or binary transformation-must be adapted to privacy-preserving settings, where neither input labels nor encoded vectors can be exposed. These operations must be embedded into the protocol logic without leaking structural information through intermediate states or memory access patterns. As a result, data representation becomes a design constraint as much as an implementation detail, influencing the feasibility and scalability of SMPC in real-world analytics pipelines.

**Secure aggregation mechanism for federated analytics**

Secure aggregation plays a central role in federated analytics settings, where data contributors independently compute local updates that are later combined into a global result. In privacy-sensitive scenarios, this aggregation must occur without revealing individual updates to any party, including the orchestrator. SMPC-based aggregation schemes address this by enabling participants to mask their local outputs with randomly generated values that cancel out upon summation [6]. These protocols allow analytics such as mean, weighted sums, or even gradient accumulation to be performed across multiple parties, with formal privacy guarantees.

One of the core advantages of this approach is its compatibility with asynchronous or partially connected systems. Participants can operate independently and submit masked results when ready, without requiring synchronized rounds or persistent connectivity. Moreover, masking techniques can be combined with cryptographic commitments or integrity checks to ensure that submitted values are structurally correct and free from tampering [7]. This makes the scheme suitable for deployment in environments such as mobile networks, industrial sensor arrays, or inter-institutional research collaborations.

The following example illustrates a simplified implementation of secure aggregation using additive masking. Each participant adds a random noise vector to their local data and distributes corresponding canceling masks to others, ensuring that individual updates remain private but the global sum remains correct.

```
import numpy as np

def generate_masks(num_parties, vector_size):
    """Generate a set of canceling masks for secure aggregation."""
    masks = np.random.randint(-10, 10, (num_parties, vector_size))
    total_mask = np.sum(masks, axis=0)
    masks[-1] -= total_mask  # Ensure masks cancel out in aggregation
    return masks

def secure_aggregate(local_updates, masks):
    """Apply masks to local updates and sum masked values."""
    masked_updates = [u + m for u, m in zip(local_updates, masks)]
    aggregate = np.sum(masked_updates, axis=0)
    return aggregate

# Simulation: three clients compute local updates
num_clients = 3
vector_dim = 5
local_updates = [np.random.randint(0, 5, vector_dim) for _ in range(num_clients)]
masks = generate_masks(num_clients, vector_dim)

# Aggregator receives masked updates
aggregated_result = secure_aggregate(local_updates, masks)
print("Securely aggregated result:", aggregated_result)
```

This example demonstrates how additive masking can enable secure aggregation in federated systems without disclosing individual data contributions [8]. The use of canceling random vectors ensures that intermediate values remain private while allowing the correct global result to be recovered. Such techniques form the foundation of many privacy-preserving analytics protocols used in cross-device, cross-organization, or cross-border data collaborations.

## Performance constraints and optimization strategies in SMPC systems

The deployment of SMPC protocols in big data contexts introduces substantial computational and communication overhead compared to conventional processing pipelines. These constraints stem from the cryptographic nature of secure computation, which often requires multiple rounds of interaction, modular arithmetic over finite fields, and the exchange of intermediate masked values. In large-scale analytics scenarios, where datasets contain millions of records or high-dimensional features, these costs can quickly render naïve implementations impractical [9]. Therefore, achieving acceptable performance in real-world applications demands both protocol-level optimizations and system-level adaptations.

One of the key performance bottlenecks lies in the network. Since many SMPC schemes rely on interactive operations between parties, latency and bandwidth become critical factors [10]. Protocols must be designed to minimize the number of communication rounds, reduce the size of transmitted payloads, and support asynchronous execution. Techniques such as precomputation, where parties compute cryptographic shares or randomness in advance, can significantly reduce online latency. In some settings, hybrid approaches that combine secure computation with differential privacy or trusted execution environments are adopted to offload expensive operations while retaining security guarantees.

At the computational level, the complexity of arithmetic operations-particularly multiplication and comparison-is another limiting factor. Secure multiplication protocols often involve multiple communication rounds or require pre-shared multiplication triples, which may not scale efficiently with the dataset size. To address this, modern SMPC frameworks implement batch processing, parallel evaluation, and optimized circuits that reduce the gate complexity of common analytical functions [11]. Additionally, approximate computation techniques, such as fixed-point arithmetic and reduced-precision encoding, are employed to strike a balance between computational efficiency and result fidelity, especially in iterative algorithms such as training machine learning models.

Resource management is also a crucial concern. SMPC implementations must be tailored to the hardware and software constraints of deployment environments, whether in cloud clusters, on-premises servers, or edge devices. Memory usage, threading, and garbage collection behavior must be optimized to prevent system stalls during execution. Moreover, adaptive scheduling mechanisms that allocate computation tasks dynamically across available nodes help improve throughput and fault tolerance. These optimizations collectively enhance the feasibility of SMPC integration into production-scale data analytics systems, ensuring that confidentiality does not come at the expense of scalability and responsiveness.

**Practical optimization strategies for scalable secure computation**

As secure multi-party computation becomes more prevalent in large-scale data analysis, practical optimization strategies play a crucial role in bridging the gap between theoretical protocols and deployable systems [12]. These techniques are aimed at mitigating specific bottlenecks inherent to secure computation, such as latency, arithmetic overhead, and limited concurrency. In real-world applications, selecting the right combination of optimizations determines not only the speed of computation but also the scalability and fault resilience of the entire system.

The following table 1 presents a summary of widely adopted optimization techniques in SMPC implementations, along with the corresponding system-level bottlenecks they address and their impact on computational performance.

Table 1

Performance optimization strategies in SMPC systems

| **Optimization technique** | **Targeted bottleneck** | **Effect on performance** |
|---|---|---|
| Precomputation of randomness | Online latency | Reduces wait time during live execution |
| Batch processing of secure operations | Per-operation overhead | Improves throughput for repeated computations |
| Use of fixed-point arithmetic | Arithmetic complexity | Decreases cost of numeric operations |
| Reduced communication rounds | Network delay | Minimizes inter-party synchronization delay |
| Parallel execution of local steps | Processing time | Accelerates computation across nodes |
| Adaptive task scheduling | Load balancing and throughput | Enhances scalability in heterogeneous systems |

The optimization techniques outlined above demonstrate how targeted improvements at both the algorithmic and infrastructural levels can significantly enhance the efficiency of SMPC-based analytics. While each method addresses a distinct performance bottleneck, their combined application enables secure computation to scale toward production-level workloads without compromising privacy guarantees. Selecting appropriate strategies requires careful evaluation of system constraints, workload characteristics, and resource availability, highlighting the need for flexible and modular SMPC frameworks tailored to real-world deployment environments.

**Integration with machine learning workflows in confidential analytics**

Integrating SMPC protocols into machine learning (ML) pipelines introduces a new layer of complexity, driven by the need to balance computational privacy with model accuracy, training efficiency, and workflow automation. In many privacy-sensitive domains-such as healthcare diagnostics, financial risk modeling, and population-level behavior prediction-ML tasks must be executed collaboratively without revealing proprietary datasets. Secure computation frameworks provide a mechanism for such privacy-preserving collaboration, allowing distributed model training or inference without centralizing data.

One of the principal challenges in SMPC-ML integration is adapting iterative optimization algorithms, such as stochastic gradient descent, to function within a secure setting. These algorithms typically require repeated computation of gradients, updates to model parameters, and aggregation of local contributions across participants. When executed under secure protocols, each of these steps becomes significantly more resource-intensive, both in terms of communication and computation. In response, research efforts have focused on optimizing sub-protocols for secure matrix operations, developing quantization-aware training methods, and reducing the depth of arithmetic circuits used in model evaluation.

Another consideration involves supporting diverse ML models, from linear classifiers to deep neural networks. While simple models can often be implemented using fixed-point arithmetic and shallow circuits, more complex architectures require approximate activation functions, layer-wise encryption, or hybrid execution models where sensitive layers are computed securely while others operate in plaintext [13]. Additionally, data preprocessing tasks-such as normalization, encoding, and feature selection-must be securely embedded into the pipeline, ensuring that privacy is maintained end-to-end.

From a systems perspective, successful integration also depends on compatibility with existing ML frameworks and infrastructure. SMPC implementations must offer clean APIs, model conversion tools, and parallelizable backends to work alongside platforms like TensorFlow, PyTorch, or federated learning engines. Scalability, fault tolerance, and reproducibility become critical, particularly in multi-organizational settings where compute environments and data schemas differ. These requirements highlight the need for robust middleware that bridges secure computation engines with modern ML ecosystems, enabling confidential analytics to transition from experimental prototypes to reliable components of production data science workflows.

**Conclusion**

The evolution of secure multi-party computation has transformed the landscape of privacy-preserving analytics, offering practical tools for collaborative computation without compromising data confidentiality. By enabling distributed entities to jointly process sensitive information while retaining control over their private inputs, SMPC addresses a critical need in data-driven sectors governed by stringent regulatory and ethical constraints. Its applicability extends beyond theoretical models, reaching real-world systems that require both analytical insight and rigorous privacy protection.

This study has examined the structural principles, protocol designs, and implementation strategies that underpin the application of SMPC in big data environments. Through practical examples, code-level demonstrations, and architectural considerations, the analysis highlights both the capabilities and limitations of current approaches. Attention was given to optimization strategies, performance bottlenecks, and integration pathways with machine learning workflows-factors that define the viability of SMPC in production-scale deployments.

As the demand for confidential analytics continues to grow, future development of SMPC systems must focus on improving computational efficiency, reducing communication overhead, and enhancing interoperability with existing data science infrastructure. The design of modular, scalable, and developer-accessible SMPC frameworks will be central to this progress, paving the way for secure and collaborative data analysis at scale.

**References**

1. Alghamdi W., Salama R., Sirija M., Abbas A.R., Dilnoza K. Secure multi-party computation for collaborative data analysis // E3S Web of Conferences. EDP Sciences. 2023. Vol. 399. P. 04034.
2. Pappa C.K. Zero-Trust Cryptographic Protocols and Differential Privacy Techniques for Scalable Secure Multi-Party Computation in Big Data Analytics // J. Electrical Systems. 2024. Vol. 20. No. 5s. P. 2114-2123.
3. Sahinbas K., Catak F.O. Secure multi-party computation-based privacy-preserving data analysis in healthcare IoT systems // Interpretable Cognitive Internet of Things for Healthcare. Cham: Springer International Publishing. 2023. P. 57-72.
4. Nookala G. Secure Multiparty Computation (SMC) for Privacy-Preserving Data Analysis // Journal of Big Data and Smart Systems. 2023. Vol. 4. No. 1.
5. Salako A.O., Adesokan-Imran T.O., Tiwo O.J., Metibemu O.C., Onyenaucheya O.S., Olaniyi O.O. Securing Confidentiality in Distributed Ledger Systems with Secure Multi-Party Computation for Financial Data Protection // Journal of Engineering Research and Reports. 2025. Vol. 27. No. 3. P. 352-373.
6. Olusegun J., Holland M., Brightwood S., Jerry H., Frank E. Distributed Secure Multi-Party Computation (SMPC) for Cloud-Based Big Data Analytics. 2024.
7. Liu T. Research on Privacy Techniques Based on Multi-Party Secure Computation // 2024 3rd International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS). IEEE. 2024. P. 912-917.
8. Liu D., Yu G., Zhong Z., Song Y. Secure multi-party computation with secret sharing for real-time data aggregation in IIoT // Computer Communications. 2024. Vol. 224. P. 159-168.
9. Ahammed M.F., Labu M.R. Privacy-preserving data sharing in healthcare: advances in secure multiparty computation // Journal of Medical and Health Studies. 2024. Vol. 5. No. 2. P. 37-47.
10. Joshi D., Sanghi A., Agarwal G., Joshi B. Techniques for Protecting Privacy in Big Data Security: A Comprehensive Review // 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT). IEEE. 2024. Vol. 1. P. 1-7.
11. Becker S., Bösch C., Hettwer B., Hoeren T., Rombach M., Trieflinger S., Yalame H. Multi-Party Computation in Corporate Data Processing: Legal and Technical Insights // Cryptology ePrint Archive. 2025.
12. Dangi D., Santhi G. Secured multi-party data release on cloud for big data privacy-preserving using fusion learning // Turkish Journal of Computer and Mathematics Education. 2021. Vol. 12. No. 3. P. 4716-4725.
13. Yogi M.K., Mundru Y. Genomic data analysis with variant of secure multi-party computation technique // Journal of Trends in Computer Science and Smart Technology. 2024. Vol. 5. No. 4. P. 450-470.